# THE INTEGRATION OF REAL DEVICE CAPABILITIES IN DISTRIBUTED APPLICATIONS BASED ON OPC TEHNOLOGY

VASILE GAITAN(1), VALENTIN POPA(1), IOAN UNGUREAN(2), NICOLETA CRISTINA GAITAN (1)
1.University "Stefan cel Mare" Suceava, 2.GENPRO 07 SRL Suceava
1.Str. Universitatii nr.13,RO-720229, 2.Bld.G. Enescu nr.38,RO-720253 Suceava
ROMANIA

*Abstract*— The aim of this paper is to present a simple yet powerful way of introducing new devices into a SCADA application based on OPC technology. In order to accomplish this, the ELECTRONIC DEVICE DESCRIPTION technology and Object Dictionary concept were used as a middleware between the SCADA application and the field devices. A text-based language was used for describing the digital communication characteristics of intelligent field instrumentation and equipment parameters – device status, diagnostic data, and configuration details – in an operating system (OS) and human machine interface (HMI) neutral environment.

*Key-Words:* - EDS, OPC, SCADA, EDDL

## 1 Introduction

At this time a big diversity of measure and control devices are used in industry: single-channel and multi-channel indicators, paperless recorders, dedicated devices, etc. All these devices can be grouped depending on the communication protocol used for the data transmissions over the industrial local networks (MODBUS, ProfiBus, CANOpen, ASCII, etc). The requests from the devices and the industrial systems regarding to flexibility (the new devices or new technology addition in an old system, the system modernized without missing its basic functionality) and production speed, increase from the viewpoint of complexity and the cost.

All information which are transmitted by these devices can be acquired on a PC with SCADA application type (which includes OPC servers and clients) [1][2][3][6][7]. In the application design process the possibility of introduction new communication protocols and new device types in the system without application rebuild must be foreseen.

## 2 Object dictionary object

At the first version of SCADA application developed inside of GenPro Company, we encountered difficulties on the introduction new device types in the system. Because of that reason, in the design process of new SCADA application version we have foreseen the elimination of these problems. This thing is quite difficult to achieve due to the big number of devices and communication protocols.

In the design process of the SCADA application we want to find a way to describe devices so that the addition of new devices into SCADA system is achieved quick and easily. After studying more industrial networks (Canopen [4], EthernetIP [5], ProfiBus[9] etc.) the use of the object dictionaries concept was chosen. Thus each device is considerate as an object collection, each object contains more data members. The access to the objects of a device is accomplished by means of the object dictionary. Each device will have an object dictionary, which helps the SCADA [6][7][8] system to access the object of the device.

Devices are considered to have 3 parts [4]:

- **The communication** – this function is provided by the communication objects and permits data transportation throughout the network;
- **Object dictionary** – represents the description of all data that can be transported throughout the network;
- **Application** – contains the functionality of the devices, with respect to interaction with the industrial process environment.

The object dictionary can be seen as an interface between the device application and the device communication functions.

The Object Dictionary is organized as an input (object) collection, looking as a table. Each entry in the object dictionary can have up-to 256 sub-inputs. Each input (object) has at least one sub-input and each input is characterized by data type and value.

The objects from object dictionary can be divided in 2 categories [4]:

- PDO Object - Process Data Object – process objects (digital input/output, analogue input etc.).
- SDO Object - Service Data Objects (address, the communication speed, the device parameters, etc.).

Object dictionary differs depending on each device types and is not stored on to device. For this reason, for each device types that are integrated in systems a description text file will be made, named EDS file. EDS is an acronym from Electronic Data Sheet and represents a text file which describes the object dictionary structure for the devices from the industrial process and which commands are used for data access for each object. Therefore, each type of device is described by the EDS file, and when it is introduced a new type of device only the EDS file is written, it is not necessary to create drivers for the devices from the local industrial networks. The introduction of a new communication protocol only involves the implementation of the acquisition module for that protocol.

The interpretation of the EDS file is made by the communication module - it reads the object dictionary structure (the number of objects, the number of sub-objects for each object, the type and the name for each object and sub-object).

The acquisition modules interpret the EDS file in order to read a minimal dictionary object structure (the objects number, the sub-objects number for each object and the type of each sub-object) and commands used for the update of each object. This information is used to update the objects and to transmit data read from devices to the communication module.

# 3 Electronic Device Description file

A text file with the "eds" extension, named Electronic Data Sheet, will be defined for each device. In this file there will be a description of the object supported by the device and the network commands which are used to interrogate the device in order to update these objects.

EDS file has the following sections:

- *FileInfo* -section which contains the information about file;
- *DeviceInfo* -section which contains the information about the device;
- *PdoObject* -section which describes the PDO objects;
- *SdoObject* -section which describes the SDO objects;
- *History* -section which describes the method for historic download;
- *Communication* -section which describes the methods used for objects upload/download.

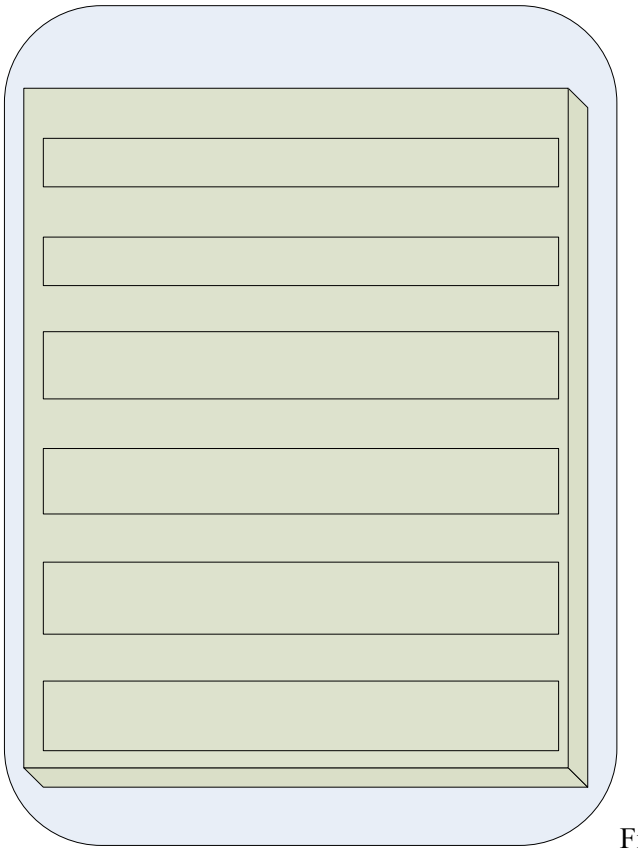

Fig. 1 Electronic device description file

## 3.1 FileInfo section

The EDS file contains information about the file, information that is used for the file version check. This information is stored in a section that is preceded by "[FileInfo]" text.

The keywords used in this section are:

*FileName* -file name;
*FileVersion* -file version (Unsigned8);
*FileRevision* -file revision (Unsigned8);
*Description* -file description (maxim 255 chars);
*CreationTime* -creation time for EDS file (the format is: hh:mm (AM|PM));
*CreationDate* - creation date for EDS file (the format is: mm-dd-YYYY);
*CreatedBy* -the name for author of this file an a *description* (maxim 255 chars);
*ModificationTime* -the time for the last modification of this file (the format is: hh:mm (AM|PM));
*ModificationDate* -the date for the last modification of this file (the format is mm-dd-YYYY);
*ModificatedBy* -the authors and description of the modification for this file (maxim 255 chars).

Example:

*[FileInfo]*
*FileName = IUM04.eds*
*FileVersion = 1*
*FileRevision = 2*
*Description = EDS for*
*Multi-channel Universal Indicator v4.0*
*CreationTime = 9.45PM*
*CreationDate = 05-05-2007*
*CreatedBy= Ioan Ungurean*
*ModificationTime = 11:30PM*
*ModificationDate = 06-06-2007*
*ModificatedBy = Ioan Ungurean*

### 3.2 DeviceInfo section

The EDS file contains particular information about the device such as:

- manufacturer name;
- manufacturer ID;
- device name.

This information can be found in DeviveInfo section. The keywords used in this section are:

*VendorName* - manufacturer name (maxim 255 chars);
*VendorNumber* - the unique ID for manufacturer (Unsigned32);
*ProductName* - device name (maxim 255 chars);
*ProductNumber* - the unique ID for device (Unsigned32);
*RevisiontNumber* - device revision (Unsigned32);

Example:
*[DeviceInfo]*
*VendorName = SIEMENS*
*VendorNumber = 1*
*ProductName = Multichannel Universal Indicator*
*ProductNumber = 1*

### 3.3 PdoObject section

In this section the PDO objects of a device are defined. The keywords used for object description are:

*ObjectName* - object name;
*AccesType* - access type for object (can take the following values: wo –write only, ro – read only, rw - read-write) ;
*DefaultValue* - it is used in order to determine the number of member for the object;
*Description* - the description for object member;
*Type* - the type for object member;
*LowLimit* - the minimal value for current member (optional, it is used only for numeric values);
*HighLimit* - the maximal value for current member (optional, it is used only for numeric values).

Example:
*[PdoObject]*
*[2001]*
*ObjectName = ValueOfChanel1*
*AccesType = ro*
*[2001sub0]*
*Description = NumberOf SubInputs*
*DefaultValue = 3*
*[2001sub1]*
*Description = StateOfChanel1*
*Type = UNSIGNED8*
*LowLimit = 0*
*HighLimit = 1*
*[2001sub2]*
*Description = The value of chanel 1*
*Type = REAL32*
*[2001sub3*
*Description = The value of chanel 1' counter*
*Type = REAL32*

### 3.4 SdoObject section

In [SdoObject] section are defined the SDO objects for device. The definitions for SDO objects are identically with definition for PDO objects.

### 3.5 History section

The procedure for history download from device is defined in the [History] section. If the device does not support the history (data logger) facility, the [History] section will not be present in EDS file. Here is the description for the history downloading procedure of the 2000 object.

*[History]*
*[2000]*
*[START_HIST]*
*FROM ModBusRegisterAddress TO ModBusRegisterAddress*
*FROM ModBusRegisterAddress TO ModBusRegisterAddress*

*[GET_INREG]*
*FROM ModBusRegisterAddress TO ModBusRegisterAddress1*

In [START_HIST] section the MODBUS registers where is transmitted data and times for the history download are defined. Also, in this section is defined the

MODBUS register which is used to read the ID for the fist record from historic. In section [GET_INREG] are defined the MODBUS register which is used to read a record from historic.
If there are more objects for which are achieved historians (data logger and event logger), the procedure for history download from a device and for these objects will be defined.

### 3.6 Communication section

In order to update the defined object's values, the commands to be sent to the device and also the answers for each command must be defined. Now, the data acquisition mode used for ModBus protocol is described. The [Communication] section contains this description.

*[IndexObiect]: FROM ModBusRegisterAddress TO ModBusRegisterAddress*

The data interpretation mode must be described too, when the acquisitioned data is interpreted. The data can be directly written in the object's memory buffer if it didn't need to be interpreted (The difference between data representation modes must be taken into account). This section is not necessary for CANOpen protocol.

As we can see in fig.2, OPC server is made of:

- OPC data server – which has implemented OPC specification interfaces.
- Communication component – which manage object dictionaries for each distinct device in system (the data read from devices are stored in a cache memory).
- Acquisition component – which reads data from devices in the system.

The EDS file interpretation is made by:

- Communication component – to read the object dictionary's structure (the number of objects, the number of sub objects belonging to an object, the type of a sub-object, objects and sub-objects name). This information is used to allocate memory for storing the data that was read from a device and also in creating the „Browse" interface by the OPC server.
- Acquisition component – in order to read a few sections of the objects dictionary (the number of objects, the number of sub objects belonging to an object, the type of a sub object), the commands that are used to update the objects, and to send the data read from a device to the communication component.

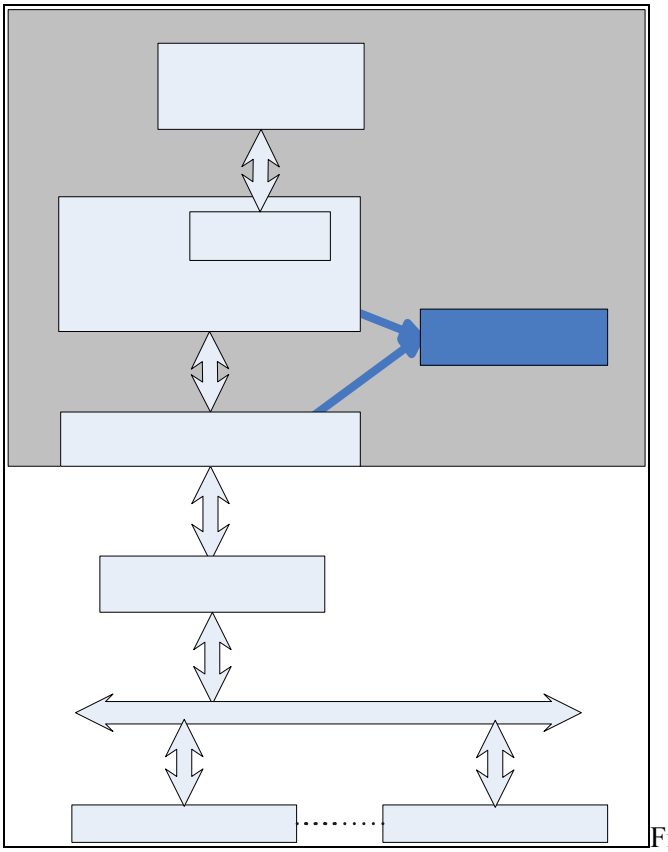

Fig.2 EDS interpretation

## 4 A practical EDS file example

We can see in figure 3 and figure 4 a practical EDS file example. If the „Intrări digitale" object initially has one member, but if it necessary to expand its size to eight members (as we can see in fig.3) the „[PdoObject]" and „[Communication]" sections of the EDS file must be modified. (as we can see in fig.4).

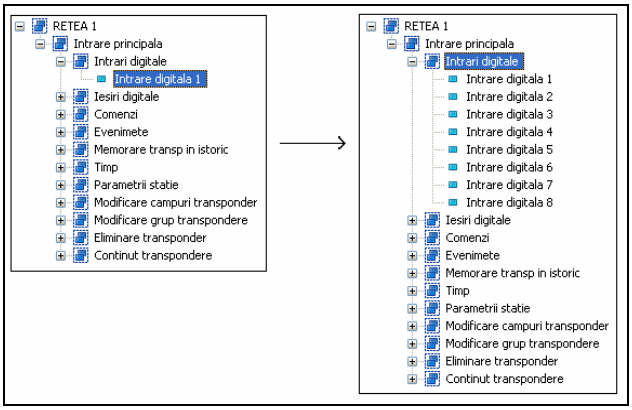


Fig.3 EDS file example

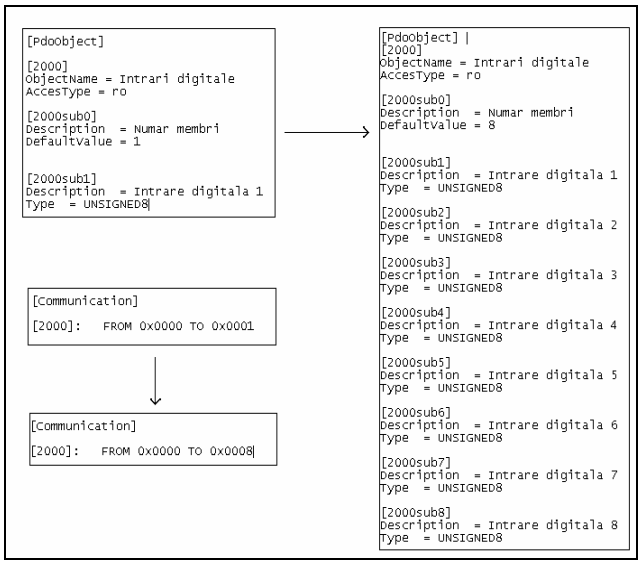

Fig.4 EDS file example

# 5 Conclusion

By using the „Electronic Device Description Language" technology a unitary way of describing devices was achieved. This description is used by the OPC server to determinate the features of each device. In order to add a new type of device a new EDS file is needed to be written and there is no need to write a new profile (this is time saving).

Modifying commands for a device implies just the editing of the EDS description file. The way of describing the object dictionary structure does not depend on the communication protocol (new protocols can be easily added).

# Acknowledgment

These researches were financed by ctr. no. 80 from 25.09.2007 (SMEDU) research grant.

*References:*

[1] Vasile GĂITAN, Cornel TURCU, Alexandru GOLOCA, Renati POPA, An RFID and OPC Technology Based Distributed System for Production Control and Monitoring, *Proceedings of the 1-st RFID Eurasia Conference*, 5-7.09.2007, Istambul, Turkey, pg. 253-258, ISBN: 978-975-01566-0-1, IEEE Catalog Number: 07EX1725, Digital Object Identifier 10.1109/RFIDEURASIA.2007.4368133, 2007

[2] www.opcfoundation.org

[3] Frank Iwanitz, Jurgen Lange (2002), *OPC Fundamentals, Implementation, and Application* 2nd rev. Ed.

[4] Olaf Pfeiffer, Andrew Ayre, Christian Keydel, (2003), *Embedded Networking with CAN and CANopen*

[5] www.odva.org

[6] Gabriel DĂNILĂ and Alexandru GOLOCA, "Creating a Transparent Interface With Field Bus Networks Using OPC Technology", vol. *Distributed Systems*, December, 2006, Suceava, Romania, ISSN/ISBN: 1842 – 68081

[7] Vasile Găitan. 2007. Using OPC technologies with the Highly Functional Distributed System, *Advances in Electrical and Computer Engineering*, Suceava, Romania 2/2003, volume 3 (10), pp. 35-46

[8] Vasile Gaitan, Cornel Turcu, Alexandru Goloca, High Complexity Control Gates with Advanced RFID Features for Production Process Monitoring, *The IEEE 22nd International Conference on Advanced Information Networking and Applications*, 25-28 march 2008, GinoWan, Okinawa, Japan, ISBN: 978-0-7695-3096-3, INSPEC Accession Number: 9912922, Digital Object Identifier: 10.1109/WAINA.2008.232, Date Published in Issue: 2008-04-03 16:08:54.0

[9] Ronaldo Hüsemann, Carlos Eduardo Pereira, A multi-protocol real-time monitoring and validation system for distributed fieldbus-based automation applications, Control Engineering Practice, Volume 15, Issue 8, August 2007, Pages 955-968